Registration No.

22829

-Technical Report-

SIMPLIFIED DYNAMICS AND MOBILITY FACTORS FOR MULTI-DISCIPLINARY OPTIMIZATION OF AN OCCUPANT CENTRIC PLATFORM

April 2012

U.S. Army Tank Automotive Research, Development, and Engineering Center
Detroit Arsenal
Warren, Michigan 48397-5000

TARDEC Technical Report No. 22829
April 2012

SIMPLIFIED DYNAMICS AND MOBILITY FACTORS FOR MULTI-DISCIPLINARY OPTIMIZATION OF AN OCCUPANT CENTRIC PLATFORM

W. Bylsma
Dynamics and Structures
U.S. Army Research, Development and Engineering Command (RDECOM)
U.S. Army Tank-automotive and Armaments Research, Development and Engineering Center (TARDEC)
Detroit Arsenal
ATTN: RDTA-RS/MS157
6501 East 11 Mile Road
Warren, Michigan 48397-5000

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)* 04-16-2012	**2. REPORT TYPE** TECHNICAL	**3. DATES COVERED** *(From - To)* 2012
4. TITLE AND SUBTITLE SIMPLIFIED DYNAMICS AND MOBILITY FACTORS FOR MULTI-DISCIPLINARY OPTIMIZATION OF AN OCCUPANT CENTRIC PLATFORM		**5a. CONTRACT NUMBER**
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) WESLEY BYLSMA		**5d. PROJECT NUMBER**
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) DYNAMICS AND STRUCTURES-US ARMY RDECOM/TARDEC ATTN: RDTA-RS/MS157 6501 E 11 MILE RD WARREN, MI 48397-5000		**8. PERFORMING ORGANIZATION REPORT NUMBER** 22829
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)		**10. SPONSOR/MONITOR'S ACRONYM(S)**
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT
Distribution Statement A: Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
This report documents the specific dynamics and mobility factors within a U.S. Army ground vehicle Design of Experiments (DOE) approach to designing an Occupant Centric Platform (OCP) using multi-disciplinary design optimization (MDO). Simplified dynamic and mobility factors that reflect pertinent vehicle design performance in these areas and that can be used in an MDO for an OCP with an underbody blast survivability focus are described. The calculation of each factor is based on portable scripts that fit well into the MDO framework.

15. SUBJECT TERMS
Occupant Centric Platform (OCP), Multi-disciplinary Design Optimization (MDO), Dynamics, Mobility, Static Stability Factor (SSF), Vehicle Cone Index (VCI)

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT Unclassified	**b. ABSTRACT** Unclassified	**c. THIS PAGE** Unclassified	Unclassified	16	Wesley Bylsma
					19b. TELEPHONE NUMBER *(include area code)* 586-282-4104

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

Contents (by Title)

SIMPLIFIED DYNAMICS AND MOBILITY FACTORS FOR MULTI-DISCIPLINARY OPTIMIZATION OF AN OCCUPANT CENTRIC PLATFORM

TARDEC Technical Report No. 22829
April 2012

1.0 INTRODUCTION

This report documents the specific dynamics and mobility factors within a U.S. Army ground vehicle Design of Experiments (DOE) approach to designing an Occupant Centric Platform (OCP) using Multi-disciplinary Design Optimization (MDO). The objection of an OCP is to "

> Understand how to balance vehicle "protection", "performance" & "payload" through an integrated survivability approach that starts with occupant protection.---[1]

As [2] states, the problem is that "We design vehicles to put Soldiers in rather than designing vehicles around Soldiers. Increasing protection levels of the platforms impacts interior volumes reducing mobility, maneuverability, and freedom of movement for occupants and leads to heavier platforms." while the challenge is to "Formulate a S&T program to make improvements to existing platforms or develop new platforms that provide appropriate increased protection from current and emerging threats and optimal space allocation for Soldiers and their gear, while decreasing platform weight and maintaining or increasing maneuverability during full spectrum operations." This encompasses a wide array of subject areas in vehicle design that includes Survivability, Mobility, Thermal, Power, Dynamics, Structures, etc. As explained in [3], MDO addresses the complex interactions between all of these areas.

> Multi-disciplinary design optimization (MDO) is a field of engineering that uses optimization methods to solve design problems incorporating a number of disciplines. As defined by Prof. Carlo Poloni, MDO is "the art of finding the best compromise". It is also known as multidisciplinary optimization and multidisciplinary system design optimization (MSDO).
>
> MDO allows designers to incorporate all relevant disciplines simultaneously. The optimum of the simultaneous problem is superior to the design found by optimizing each discipline sequentially, since it can exploit the interactions between the disciplines. However, including all disciplines simultaneously significantly increases the complexity of the problem.
>
> These techniques have been used in a number of fields, including automobile design, naval architecture, electronics, [[architecture], computers, and electricity distribution. However, the largest number of applications have been in the field of aerospace engineering, such as aircraft and spacecraft design. For example, the proposed Boeing blended wing body (BWB) aircraft concept has used MDO extensively in the conceptual and preliminary design stages. The disciplines considered in the BWB design are aerodynamics, structural analysis, propulsion, control theory, and economics.---[3]

A wide range of software tools are unique to each discipline. Several MDO software packages are available, see [4] and [5] as examples, that provide solutions for handling and incorporating the results from each. As stated in [4], these are "... used to combine cross-disciplinary models and applications together in a simulation process flow, automate their execution, explore the resulting design space, and identify the optimal design parameters subject to required constraints." Within this framework the initial simulation complexity can be reduced by using simplified models. As optimized results are received, refinements can be made to adjust resolution as required.

The objective of this report is to provide simplified dynamic and mobility factors that reflect pertinent vehicle design performance in these areas and that can be used in an MDO for an OCP with an underbody blast survivability focus. The calculation of each factor is described below and based on portable scripts that fit well into the framework described above.

2.0 DYNAMICS

The end result of neglecting safety can be as consequential as survivability threats. Resistance to vehicle rollover, both military as well as civilian, has to be considered in vehicle design. While rollover is a dynamic phenomenon, simplified measures can be used to gauge the rollover propensity. One of these is the Static Stability Factor (SSF). As noted from [6],

> "The Static Stability Factor (SSF) of a vehicle is an at-rest calculation of its rollover resistance based on its most important geometric properties. SSF is a measure of how top-heavy a vehicle is.
>
> A vehicle's SSF is calculated using the formula SSF=T/2H, where T is the "track width" of the vehicle and H is the "height of the center of gravity" of the vehicle. The track width is the distance between the centers of the right and left tires along the axle. The location of the center of gravity is measured in a laboratory to determine the height above the ground of the vehicle's mass. The lower the SSF number, the more likely the vehicle is to roll over in a single-vehicle crash."

A description of the specific vehicle parameters described above are shown in Figure 1.

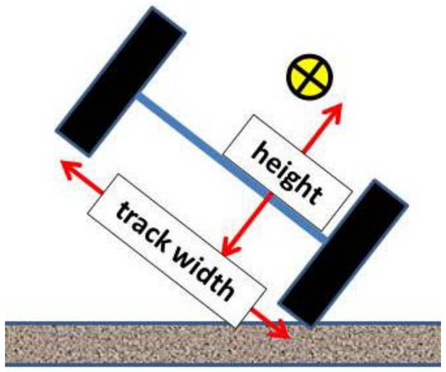


Figure 1 – Static Stability Factors

From [6] further details are given that "The SSF metric is based on a rigid-body model of a vehicle sliding laterally on a surface. For such a model, the point of incipient rollover occurs when the sum of the lateral forces divided by the weight of the vehicle, W, is greater than the SSF: sum of lateral forces/W > SSF (= T/2H). For a vehicle to roll over, the lateral forces must be sustained for a sufficient period of time." Findings from [6] also indicate that the SSF's relevance to rollover is "based on the laws of physics and captures important vehicle characteristics related to rollover" while "metrics derived from dynamic testing are needed to complement static measures”. From these conclusions SSF is determined to be a satisfactory simplified model to predict one important element of dynamics that relates to OCP design parameters.

2.1 SSF CALCULATION

A Python (see [12]) script was developed to read an input file of variable/value pairs, calculate the SSF, and write out a file of the variable/value pairs including the SSF factor (see Appendix A.1 – latw.py). Including the input factors in the output file provides context for the output calculation and can help diagnose errors during initial development. Reading an input file and writing an output file allows the script to be easily adapted to any MDO software package described in section 1.0.

The SSF is returned from a call to the dynutil library (see Appendix A.3 – dynutil.py) as

```
ssf = dynutil.calc_SSF(v_t, v_h)
```

and defined as

$$SSF = \frac{t}{2h}$$

where higher values represent better stability (The HMMWV is around 1.0). Once the input file has been created the script is called as “`latw.py [input file] [output file]`”. An example is shown below.

```
latw.py latin.w latout.w
```

2.2 SSF INPUT FILE

The format of the input file is simply variable/value pairs, one set per line, separated by tabs or spaces. An example is shown below.

latin.w

```
#================================================================================
#
#       Description:
#               Input file defining variable values to Calculate Static Stability
#               Factor (SSF)
#
#================================================================================
# t             # track width [in.]
# h             # CG height above ground [in.]
#================================================================================
# use tab or space separators
#================================================================================
t                       76
h                       36
```

2.3 SSF OUTPUT FILE

The format of the output file is simply variable/value pairs, one set per line, separated by tabs. An example is shown below.

latout.w

```
h       36.000000
t       76.000000
SSF     1.055556
```

3.0 MOBILITY

There is no more direct measure of mobility than the determination of whether a vehicle becomes immobile---it cannot move. This is done by comparing the available traction to total resistance. In a cross-country setting, grades (longitudinal slope), side slope, engine power, etc. are several contributing factors but soil strength is the key driver. The immobile or “NO-GO” area is demonstrated through the use of the mobility map and speed profile, in Figure 2, where the speed falls below the average speed for less than 100% of the terrain area. Soil specific details are outlined in [8] where the beginnings of soil-vehicle models are discussed along with several metrics for directly quantifying the ability of vehicles to traverse soft-soil terrain. One of these is Vehicle Cone Index (VCI). As noted from [9],

> VCI is defined as the minimum soil strength necessary for a self-propelled vehicle to consistently make a prescribed number of passes in track without becoming immobilized. Historical testing usually focused on measuring the minimum soil strength required for a vehicle to make one pass (VCI1) and/or 50 passes (VCI50). Using VCI measurements for a number of different vehicles, the US Army ERDC developed a set of empirical equations that predict VCI1 and VCI50 from relatively simple vehicle characteristics (i.e., weight and running-gear dimensions).

A description of the specific vehicle parameters that influence the empirical relationships are described in Figure 3 [9].

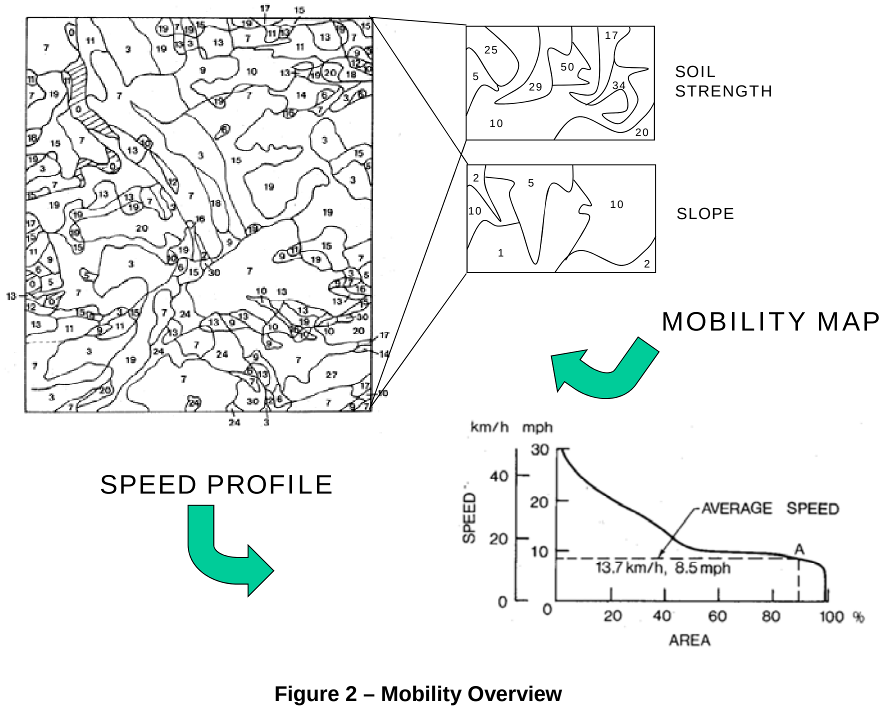
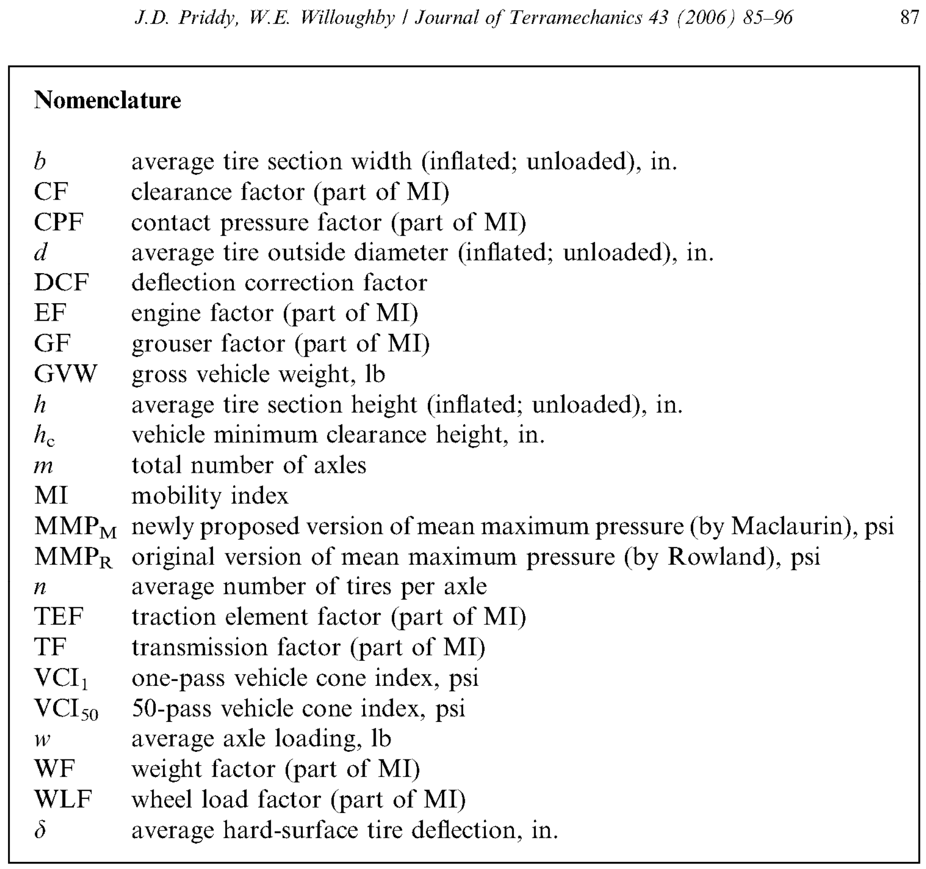


Figure 2 – Mobility Overview



J.D. Priddy, W.E. Willoughby / Journal of Terramechanics 43 (2006) 85–96 87

Nomenclature

b	average tire section width (inflated; unloaded), in.
CF	clearance factor (part of MI)
CPF	contact pressure factor (part of MI)
d	average tire outside diameter (inflated; unloaded), in.
DCF	deflection correction factor
EF	engine factor (part of MI)
GF	grouser factor (part of MI)
GVW	gross vehicle weight, lb
h	average tire section height (inflated; unloaded), in.
h_c	vehicle minimum clearance height, in.
m	total number of axles
MI	mobility index
MMP_M	newly proposed version of mean maximum pressure (by Maclaurin), psi
MMP_R	original version of mean maximum pressure (by Rowland), psi
n	average number of tires per axle
TEF	traction element factor (part of MI)
TF	transmission factor (part of MI)
VCI_1	one-pass vehicle cone index, psi
VCI_{50}	50-pass vehicle cone index, psi
w	average axle loading, lb
WF	weight factor (part of MI)
WLF	wheel load factor (part of MI)
δ	average hard-surface tire deflection, in.

Figure 3 – Vehicle Cone Index Factors

From [10], The International Society for Terrain-Vehicle Systems (ISTVS) defines VCI as one of its standard terms. Its usage is also described in [11]. From these conclusions VCI is determined to be a satisfactory simplified model to predict one important element of mobility that relates to OCP design parameters.

3.1 VCI CALCULATION

A Python (see [12]) script was developed to read an input file of variable/value pairs, calculate the VCI, and write out a file of the variable/value pairs including the VCI factor (see Appendix A.2 – vciw.py). As sated in section 2.1, including the input factors in the output file provides context for the output calculation and can help diagnose errors during initial development. Reading an input file and writing an output file allows the script to be easily adapted to any MDO software package described in section 1.0.

The VCI is returned from a call to the dynutil library (see Appendix A.3 – dynutil.py) as

```
vci1 = dynutil.calc_VCI1(v_gvw, v_m, v_b, v_d, v_h, v_hc, v_n, v_delt, v_CGF, v_CEF, v_CTF)
```

and defined as

Vehicle Cone Index (one-pass, wheeled vehicle)

$$\mathrm{VCI_1} = \mathrm{f(MI, DCF)}$$

$$\text{where } \begin{cases} \mathrm{MI} \leq 115 \rightarrow \mathrm{VCI_1} = \left(11.48 + 0.2\mathrm{MI} - \dfrac{39.2}{\mathrm{MI} + 3.74}\right)\mathrm{DCF}, \\ \mathrm{MI} > 115 \rightarrow \mathrm{VCI_1} = (4.1\mathrm{MI}^{0.446})\mathrm{DCF}. \end{cases}$$

Mobility Index

$$\mathrm{MI} = \left(\frac{\mathrm{(CPF)(WF)}}{\mathrm{(TEF)(GF)}} + \mathrm{WLF} - \mathrm{CF}\right)\mathrm{(EF)(TF)}$$

Mobility Index Factors

$$\mathrm{CPF} = \frac{\mathrm{w}}{0.5\mathrm{ndb}}$$

$$\mathrm{TEF} = \frac{10 + \mathrm{b}}{100}$$

$$\mathrm{WLF} = \frac{\mathrm{w}}{2000}$$

$$\mathrm{CF} = \frac{\mathrm{h_c}}{10}$$

$\mathrm{GF} = 1 + 0.05\mathrm{C_{GF}}$, where $\mathrm{C_{GF}} = 1$ if tire chains are used or 0 if not
$\mathrm{EF} = 1 + 0.05\mathrm{C_{EF}}$, where $\mathrm{C_{EF}} = 1$ if $\mathrm{PWR} < 10\,\dfrac{\mathrm{hp}}{\mathrm{ton}}$ or 0 if not
$\mathrm{TF} = 1 + 0.05\mathrm{C_{TF}}$, where $\mathrm{C_{TF}} = 1$ if manual transmission or 0 if automatic

$$\mathrm{WF} = \mathrm{C_{WF1}}\frac{\mathrm{w}}{1000} + \mathrm{C_{WF2}},$$

$$\text{where } \begin{cases} \mathrm{w} < 2000\ lbs. \rightarrow \mathrm{C_{WF1}} = 0.553 \text{ and } \mathrm{C_{WF2}} = 0, \\ 2000 \leq \mathrm{w} < 13{,}500\ lbs. \rightarrow \mathrm{C_{WF1}} = 0.033 \text{ and } \mathrm{C_{WF2}} = 1.050, \\ 13{,}500 \leq \mathrm{w} < 20{,}000\ lbs. \rightarrow \mathrm{C_{WF1}} = 0.142 \text{ and } \mathrm{C_{WF2}} = -0.420, \\ 20{,}000 \leq \mathrm{w} < 31{,}500\ lbs. \rightarrow \mathrm{C_{WF1}} = 0.278 \text{ and } \mathrm{C_{WF2}} = -3.115, \\ 31{,}500 \leq \mathrm{w} \rightarrow \mathrm{C_{WF1}} = 0.836 \text{ and } \mathrm{C_{WF2}} = -20.686. \end{cases}$$

Deflection Correction Factor

$$\mathrm{DCF} = \left(\frac{0.15}{\delta/\mathrm{h}}\right)^{0.25}$$

where lower values represent better mobility (The original HMMWV is around 25.0). Once the input file has been created the script is called as "`vciw.py [input file] [output file]`". An example is shown below.

```
vciw.py vciin.w vciout.w
```

3.2 VCI INPUT FILE

The format of the input file is simply variable/value pairs, one set per line, separated by tabs or spaces. An example is shown below.

vciin.w

```
#================================================================================
#
#       Description:
#               Input file defining variable values to Calculate One Pass Vehicle Cone
#               Index (VCI) for Wheeled Vehicles using function calc_VCI1
#
#================================================================================
# gvw           # Gross Vehicle Weight [lbs.]
# m             # total number of axles
# b             # average tire section width (inflated, unloaded) [in.]
# d             # average tire outside diameter (inflated, unloaded), [in.]
# h             # average tire section height (inflated, unloaded) [in.]
# hc            # vehicle minimum clearance height [in.]
# n             # average number of tires per axle
# delt  # average hard-surface tire deflection [in.]
# CGF           # 1 if tire chains are used or 0 if not
# CEF           # 1 if PWR (power to weight ratio) < 10 [hp/ton] or 0 if not
# CTF           # 1 if manual transmission or 0 if automatic
#================================================================================
# use tab or space separators
#================================================================================
gvw             15000
m                       2
b                       14
d                       40
h                       10
hc                      15
n                       2
delt            1
CGF             0
CEF             0
CTF             0
service0 278%
```

3.3 VCI OUTPUT FILE

The format of the output file is simply variable/value pairs, one set per line, separated by tabs. An example is shown below.

vciout.w

```
CEF     0.000000
CGF     0.000000
CTF     0.000000
b       14.000000
d       40.000000
delt    1.000000
gvw     15000.000000
h       10.000000
hc      15.000000
m       2.000000
n       2.000000
VCI1    28.675231
```

4.0 SUMMARY

A description of simplified dynamic (SSF) and mobility (VCI) metrics relating to vehicle design parameters for MDO of an OCP have been described. Portable Python scripts, able to perform the calculation of SSF and VCI on a wide array of computing platforms, are provided in the Appendix. The use of ASCII format (see [13]) input/output files readily accommodates the writing of parameter data values and reading of output metric values necessary for automation of the simulation process flow by MDO software.

REFERENCES

[1] Rodgers, Paul. "Occupant Centric Integrated Survivability". Presented at the Armored Vehicle Survivability Conference, Munich, Germany, December 2009. (http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA513227).

[2] Freeman, Marilyn. "Army Science and Technology Top Ten Challenges". Association of the United States Army (AUSA). October 2011. (https://www.alt.army.mil/portal/page/portal/oasaalt/documents/Army_Science_and_Technology_Top_Ten_Challenges.pdf

[3] Multi-disciplinary Design Optimization Definition (http://en.wikipedia.org/wiki/Multidisciplinary_design_optimization).

[4] Isight. Dassault Systemes (http://www.3ds.com/products/simulia/portfolio/isight-simulia-execution-engine/overview/)

[5] modeFRONTIER. Esteco. (http://www.esteco.com/home/mode_frontier/mode_frontier.html)

[6] National Highway Transportation Safety Administration (NHTSA). (www.nhtsa.dot.gov)

[7] Transportation Research Board. "The National Highway Traffic Safety Administration's rating system for rollover resistance: an assessment". Special Report 265. ISBN 0-309-07249-2. (http://onlinepubs.trb.org/onlinepubs/sr/sr265.pdf).

[8] Rula, Nuttall. "An Analysis of Ground Mobility Models (ANAMOB). Technical Report M-71-4, July 1971. U.S. Army Engineer Waterways Experiment Station, Vicksburg, MS. Defense Technical Information Center (DTIC), AD 886 513, Approved for public release, distribution unlimited.

[9] Priddy, Willoughby. "Clarification of vehicle cone index with reference mean maximum pressure". Journal of Terramechanics 43 (2006) 85-96. (https://www.elsevier.com/locate/jterra).

[10] International Society for Terrain-Vehicle Systems (ISTVS) Standards. Journal of Terramechanics Vol. 14, No. 3, pp. 153 to 182 (1977). Pergamon Press (Printed in Great Britain).

[11] Wong, J. Y. Theory of Ground Vehicles (4th Edition). Wiley (August 2008). ISBN-10: 0-470-17038-7.

[12] Python Programming Language. (http://www.python.org).

[13] American Standard Code for Information Interchange (ASCII). (http://en.wikipedia.org/wiki/ASCII)

APPENDIX A.1 – LATW.PY

```
#!/usr/bin/python

import sys
import re
import dynutil

#================================================================================
#       Description:
#               latw.py [input file] [output file]
#
#               Reads an input file of variable/value pairs, calculates SSF, and
#               writes an output file of the variable/value pairs, including SSF.
#
#       Input File Format Example:
#
#|#==============================================================================
#|# t           # track width [in.]
#|# h           # CG height above ground [in.]
#|#==============================================================================
#|# use tab or space separators
#|#==============================================================================
#|t                     76
#|h                     36
#
#       Output File Format Example:
#
#|h     36.000000
#|t     76.000000
#|SSF   1.055556
#
#================================================================================
#
#       Modified:
#               2012-04-05      Initial
#
#================================================================================
#
#       References:
#               1)      see dynutil.py
#
#================================================================================
#
#       Variables:
#               fi                      input file name
#               fo                      output file
#
#               arr             list of variable and values per line
#               vars            dictionary of variable/value pairs
#               vars_num        minimum number of variable/value pairs needed
#               cnt             count of variable/value pairs obtained
#
#================================================================================

if (len(sys.argv) < 3):
        print "Usage:  latw.py [input file] [output file]"
        sys.exit()

fi = sys.argv[1]
fo = sys.argv[2]
arr = []
vars = {}
vars_num = 2
cnt = 0

#================================================================================
#       Process input file
#================================================================================
fin = open(fi,'r')

for line in fin:
        l = line.strip("\n")
        arr = re.split("[ \t]*",l)

        if (arr[0][0] == "#"):
                continue
        if (len(arr) < 2):
                print "Error in Variable/Value pairs."
                sys.exit()
        else:
                vars[arr[0]] = float(arr[1])
                cnt = cnt + 1

fin.close()

if (cnt < vars_num):
        print "Only %d Variable/Value pairs found, need %d." % (cnt,vars_num)
        sys.exit()

#================================================================================
#       Process output file
#================================================================================

fon = open(fo,'w')

for key in sorted(vars.iterkeys()):
        print >> fon,"%s\t%f" % (key,vars[key])

v_t = vars["t"]
v_h = vars["h"]

ssf = dynutil.calc_SSF(v_t, v_h)
print >> fon,"%s\t%f" % ("SSF",ssf)

fon.close()

#================================================================================
```

APPENDIX A.2 – VCIW.PY

```
#!/usr/bin/python

import sys
import re
import dynutil

#================================================================================
#	Description:
#		vciw.py [input file] [output file]
#
#		Reads an input file of variable/value pairs, calculates VCI1, and
#		writes an output file of the variable/value pairs, including VCI1.
#
#	Input File Format Example:
#
#|#================================================================================
#|# gvw		# Gross Vehicle Weight [lbs.]
#|# m			# total number of axles
#|# b			# average tire section width (inflated, unloaded) [in.]
#|# d			# average tire outside diameter (inflated, unloaded), [in.]
#|# h			# average tire section height (inflated, unloaded) [in.]
#|# hc		# vehicle minimum clearance height [in.]
#|# n			# average number of tires per axle
#|# delt			# average hard-surface tire deflection [in.]
#|# CGF		# 1 if tire chains are used or 0 if not
#|# CEF		# 1 if PWR (power to weight ratio) < 10 [hp/ton] or 0 if not
#|# CTF		# 1 if manual transmission or 0 if automatic
#|#================================================================================
#|# use tab or space separators
#|#================================================================================
#|gvw		15000
#|m			2
#|b			14
#|d			40
#|h			10
#|hc			15
#|n			2
#|delt		1
#|CGF		0
#|CEF		0
#|CTF		0
#
#	Output File Format Example:
#
#|CEF		0.000000
#|CGF		0.000000
#|CTF		0.000000
#|b		14.000000
#|d		40.000000
#|delt	1.000000
#|gvw		15000.000000
#|h		10.000000
#|hc		15.000000
#|m		2.000000
#|n		2.000000
#|VCI1	28.675231
#
#================================================================================
#
#	Modified:
#		2012-04-05	Initial
#
#================================================================================
#
#	References:
#		1)	see dynutil.py
#
#================================================================================
#
#       Variables:
#               fi              input file name
#               fo              output file
#
#     arr      list of variable and values per line
#     vars     dictionary of variable/value pairs
#     vars_num minimum number of variable/value pairs needed
#     cnt      count of variable/value pairs obtained
#
#===============================================================================

if (len(sys.argv) < 3):
   print "Usage:  vciw.py [input file] [output file]"
   sys.exit()

fi = sys.argv[1]
fo = sys.argv[2]
arr = []
vars = {}
vars_num = 11
cnt = 0

#===============================================================================
#       Process input file
#===============================================================================

fin = open(fi,'r')

for line in fin:
        l = line.strip("\n")
        arr = re.split("[ \t]*",l)

        if (arr[0][0] == "#"):
                continue
        if (len(arr) < 2):
                print "Error in Variable/Value pairs."
                sys.exit()
        else:
                vars[arr[0]] = float(arr[1])
                cnt = cnt + 1

fin.close()

if (cnt < vars_num):
        print "Only %d Variable/Value pairs found, need %d." % (cnt,vars_num)
        sys.exit()

#===============================================================================
#       Process output file
#===============================================================================

fon = open(fo,'w')

for key in sorted(vars.iterkeys()):
        print >> fon,"%s\t%f" % (key,vars[key])

v_gvw = vars["gvw"]
v_m = vars["m"]
v_b = vars["b"]
v_d = vars["d"]
v_h = vars["h"]
v_hc = vars["hc"]
v_n = vars["n"]
v_delt = vars["delt"]
v_CGF = vars["CGF"]
v_CEF = vars["CEF"]
v_CTF = vars["CTF"]

vci1 = dynutil.calc_VCI1(v_gvw, v_m, v_b, v_d, v_h, v_hc, v_n, v_delt, v_CGF, v_CEF, v_CTF)
print >> fon,"%s\t%f" % ("VCI1",vci1)
```

```
fon.close()

#================================================================================
```

APPENDIX A.3 – DYNUTIL.PY

```
#================================================================================
#
def calc_VCI1(gvw, m, b, d, h, hc, n, delt, CGF, CEF, CTF):
#
#================================================================================
#
#       Description:
#               Calculate One Pass Vehicle Cone Index (VCI) for Wheeled Vehicles
#               (Lower numbers represent better mobility.  Less than 20 is good.)
#
#================================================================================
#
#       Modified:
#               2012-04-05      Initial
#
#================================================================================
#
#       References:
#               1)      Annex E - HMMWV MECV Performance Specification
#
#               2)      Clarification of vehicle cone index w/ref to MMP
#                       Journal of Terramechanics 43 (2006) 85-96
#                       J.  D. Priddy, W. E. Willoughby
#                       (www.elsevier.com/locate/jterra)
#
#================================================================================
#
#       Variables:
#               gvw     Gross Vehicle Weight [lbs.]
#               m               total number of axles
#               b               average tire section width (inflated, unloaded) [in.]
#               d               average tire outside diameter (inflated, unloaded), [in.]
#               h               average tire section height (inflated, unloaded) [in.]
#               hc              vehicle minimum clearance height [in.]
#               n               average number of tires per axle
#               delt    average hard-surface tire deflection [in.]
#               CGF     1 if tire chains are used or 0 if not
#               CEF     1 if PWR (power to weight ratio) < 10 [hp/ton] or 0 if not
#               CTF     1 if manual transmission or 0 if automatic
#
#               w               average axle loading [lbs.] = gvw / m
#               CPF     contact pressure factor
#               TEF     Traction Element Factor
#               WLF     Weight Load Factor
#               CF              Clearance Factor
#               GF              Grouser Factor
#               EF              Engine Factor
#               TF              Transmission Factor
#               WF              Weight Factor
#               DCF     Deflection Correction Factor
#               MI              Mobility Index
#               VCI1    Vehicle Cone Index (One Pass)
#
#================================================================================

        w = gvw / m

# Mobility Index Factors
        # Contact Pressure Factor (CPF)
        CPF = w / (0.5 * n * d * b)

        # Traction Element Factor (TEF)
        TEF = (10.0 + b) / 100.0

        # Weight Load Factor (WLF)
        WLF = w / 2000.0

        # Clearance Factor (CF)
        CF = hc / 10.0

        # Grouser Factor (GF)
        GF = 1 + 0.05 * CGF

        # Engine Factor (EF)
        EF = 1 + 0.05 * CEF

        # Transmission Factor (TF)
        TF = 1 + 0.05 * CTF

        # Weight Factor (WF)
        if (w < 2000.0):
                CWF1 = 0.553
                CWF2 = 0.0
        elif (2000.0 <= w and w < 13500.0):
                CWF1 = 0.033
                CWF2 = 1.05
        elif (13500.0 <= w and w < 20000.0):
                CWF1 = 0.142
                CWF2 = -0.42
        elif (20000.0 <= w and w < 31500.0):
                CWF1 = 0.278
                CWF2 = -3.115
        elif (31500.0 <= w):
                CWF1 = 0.836
                CWF2 = -20.686
        WF = CWF1 * (w / 1000.0) + CWF2

        # Deflection Correction Factor (DCF)
        DCF = (0.15 / (delt / h)) ** 0.25

        # Mobility Index (MI)
        MI = (((CPF * WF) / (TEF * GF)) + WLF - CF) * EF * TF

        # Vehicle Cone Index (One Pass)
        if (MI <= 115.0):
                VCI1 = (11.48 + 0.2 * MI - (39.2 / (MI + 3.74))) * DCF
        elif (MI > 115.0):
                VCI1 = (4.1 * MI ** 0.446) * DCF

        return VCI1

#===============================================================================
#
def calc_SSF(t, h):
#
#===============================================================================
#
#       Description:
#               Calculate Static Stability Factor (SSF)
#               (Higher values represent better stability.  1.0 is good.)
#
#===============================================================================
#
#       Modified:
#               2012-04-05      Initial
#
#===============================================================================
#
#       References:
#               1) Special Report 265 - "The National Highway Traffic Safety
#                       Administration's rating system for rollover resistance : an
#                       assessment", Transportation Research Board (TRB).  ISBN 0-309-07249-2
#
#               2)      http://www.nhtsa.gov
#
#               3)      http://www.safetyresearch.net/safety-issues/rollover-stability
#
#===============================================================================
#
```

```
#       Variables:
#               t               track width [in.]
#               h               CG height above ground [in.]
#
#               SSF     Static Stability Factor
#
#================================================================================

        # Static Stability Factor (SSF)
        SSF = t / (2 * h)

        return SSF

#================================================================================
```